Foreword

C++/CLI was originally envisioned as a high-level assembler for the .NET runtime, much like C is often considered a high-level assembler for native code generation. That original vision even included the ability to directly mix in IL with C++ code, mostly eliminating the need for the IL assembler `ilasm`.

As the design of C++/CLI evolved, this vision was scaled back. We, the Microsoft C++ team, still wanted C++/CLI to be a systems programming language for .NET, but we decided that exposing the full capabilities of the CLR to other languages wasn't a good idea. After all, language interoperability is a significant factor in the success of .NET.

We knew C++ programmers would expect powerful features unavailable in other languages, like C#, so we decided exposing common C++ constructs in a familiar manner when using C++/CLI was critical. For example, the RAII idiom frequently used in ISO standard C++ programs needed to work in a similar syntactic manner when using a reference type. Likewise, programmers expected templates to work seamlessly with reference types, value types, and interfaces.

We were a bit surprised that programmers found C++/CLI's predecessor Managed C++ unacceptably ugly. We thought new keywords should have leading double underscores, because that was the way to add such extensions while conforming to standard C++. Early in the C++/CLI design process, we looked for ways to make the language look nice and still contain strictly conforming extensions. While contextual and whitespace keywords are a little unconventional (and a bit of a pain to implement), they certainly make C++/CLI look much nicer and give it that first-class feel we were looking for.

In the end, I think we found the right balance between C++ power and familiarity and a clean syntax to enable access to the .NET runtime. Hopefully, you'll enjoy using C++/CLI as much as we enjoyed creating it. After working with Dean on the Visual C++ compiler front end, I know you'll find his insights and explanations valuable as you learn, and eventually master, C++/CLI.

Jason Shirk
Software Development Engineer
Visual C++ Front End
Microsoft Corporation